REVISTA BRASILEIRA DE MECATRÔNICA
FACULDADE SENAI DE TECNOLOGIA MECATRÔNICA

AS MÉTRICAS DE SOFTWARE EM FAVOR DO DESENVOLVEDOR COMO FERRAMENTAS PARA A PRODUTIVIDADE E A QUALIDADE EM SOLUÇÕES EMBARCADAS

SOFTWARE METRICS AS A VALUABLE PRODUCTIVITY TOOL FOR DEVELOPMENT OF QUALITY EMBEDDED SOLUTIONS

Luiz Carlos Canno[1], [iii]
Leandro Poloni Dantas[2], [ii]
Fernando Simplicio de Sousa[3], [iii]
Marcones Cleber Brito da Silva[4], [iv]

Data de submissão: (26/05/2023) Data de aprovação: (7/8/2023)

RESUMO

Muitos profissionais da área de software concordam que um atributo importante dos sistemas embarcados é a qualidade, traduzida como a menor incidência de defeitos quanto possível. No entanto as empresas de desenvolvimento destes softwares precisam ao mesmo tempo aprimorar seus níveis de produtividade sem esquecer a qualidade do produto e uma forma efetiva e prática para medir esses aprimoramentos. Assim esta pesquisa tem por finalidade orientar as pessoas envolvidas no arcabouço do projeto, em especial os desenvolvedores de software no estabelecimento de metas para alcançar melhores padrões de qualidade e produtividade durante o processo de desenvolvimento do produto de sistemas embarcados que são implantados através de um processo exaustivo das análises de requisitos. Em especial aqueles ditos não funcionais, deste modo, estes sistemas precisam de um teste completo e processo de verificação e validação.

Palavras-chave: métricas de software; software embarcado; qualidade e produtividade de software; engenharia de requisitos.

ABSTRACT

There is a consensus amongst professionals in the area of software development that quality is an important attribute which leads to fewer errors. Software developing companies engaged in improving productivity levels without letting quality lag should utilize a practical and effective process to measure this improvement. This present study aims to guide project stakeholders, namely software developers, in the direction of establishing metrics to improve

[1] Professor Esp. na Faculdade de Tecnologia SENAI São Paulo – Campus "Anchieta". E-mail: luis.canno@sp.senai.br.
[2] Professor Dr. na Faculdade de Tecnologia SENAI São Paulo – Campus "Anchieta". E-mail: leandro.poloni@sp.senai.br.
[3] Professor Me. na Faculdade de Tecnologia SENAI São Paulo – Campus "Anchieta". E-mail: fernando.simplicio@sp.senai.br.
[4] Professor Me. na Faculdade de Tecnologia SENAI São Paulo – Campus "Anchieta". E-mail: marcones.silva@sp.senai.br.

productivity and quality standards during the embedded software development cycle through an exhaustive process of requirements analysis; mainly those deemed not-functional, which require a thorough verification and validation process.

Keywords: software metrics; embedded software; software quality and productivity; requirements engineering.

1 INTRODUÇÃO

Ouve-se com frequência sobre o termo “software” e, portanto entendem-se numa visão restritiva que softwares são programas de computador, no entanto na afirmação de Sommerville (2011), software não é apenas um programa, mas também toda a documentação associada e os dados de configuração necessários para fazer com que esses programas operem corretamente. Faz-se aqui uma distinção entre softwares desenvolvidos como aplicativos de computador e os sistemas computacionais embarcados, o primeiro, em sua maioria funciona numa base dependente de outros softwares, o segundo, por sua vez é independente e tem a aptidão de exercer uma única função tal qual sua especificação.

1.1 Problema de pesquisa

O desenvolvimento de software embarcado possui algumas peculiaridades, que sob alguns aspectos podem diferenciar-se do processo clássico de desenvolvimento de outros tipos de softwares, dentre eles destacam-se as condições relativas a um grande número de interdependências, tanto no que se refere a requisitos de solução em nível de programação como as necessidades proeminentes do hardware e seus periféricos, ou seja, o software deve ter objetivos diretos voltados à funcionalidade, bem como aos recursos de hardware disponíveis visando tarefas específicas (Gomes, 2010).

Desta maneira, as métricas podem ser importantes aliadas dos desenvolvedores, a que possam traduzir um nível de excelência a confiabilidade para verificação e validação a cada etapa do desenvolvimento, onde os testes do sistema revelarão a incidência de erros que devem ser tratados. Este trabalho pretende investigar, dentro de um período não abalizado, fundamentações com bases de pesquisas em literaturas consagradas por autores experientes, levando-se em consideração apenas e tão somente às métricas de software pertinentes à produtividade e qualidade do software.

1.2 Objetivo

O objetivo desta pesquisa foi buscar as relações desde os requisitos dos modelos clássicos, alinhado aos métodos automáticos que contribuíssem em propostas de soluções aos problemas de gestão enfrentados no desenvolvimento de sistemas embarcados pelos profissionais dedicados à engenharia de software, com sugestões das boas práticas que permeiam as métricas relacionadas à qualidade e a produtividade.

Com as definições destes propósitos, o desenvolvimento deste trabalho envolveu uma pesquisa investigativa em publicações cujos tópicos relacionam-se com as práticas tradicionais da engenharia de software, e correlações com as soluções embarcadas onde os propósitos convergem com a comparação, à explicitação e demonstração da temática tratada pelos autores.

1.3 Justificativa

Como a computação ubíqua (invisível) se torna uma realidade no cotidiano, as demandas por produtividade e qualidade de softwares embarcados crescem incessantemente. A fim de lidar com esta pressão, os desenvolvedores de software procuram novas abordagens para gerenciar o ciclo de desenvolvimento, visando o cumprimento dos prazos dentro do orçamento e livre de falhas. A previsão dos defeitos de software tem foco em como reduzir os custos de testes bem como melhorar a qualidade do produto final. O tema, portanto, foi selecionado pela importância de suas técnicas e ações previstas nas atribuições do profissional desenvolvedor de sistemas embarcados.

2 REVISÃO DE LITERATURA

Para exemplificar como definir uma metodologia dinâmica que corrobore eficácia aos processos de produtividade e qualidade no desenvolvimento do software Pfleeger (2007), afirma que o início de cada processo dentro do ambiente de desenvolvimento visa naturalmente à identificação dos requisitos que possam atender as expectativas dos clientes como também dos usuários do sistema.

De acordo com Sommerville (2011), não há maneira simples de fazer uma estimativa ao mesmo tempo precoce e precisa, do esforço necessário para desenvolver um sistema de software. As estimativas iniciais podem ter de serem feitas com base num nível elevado quando se consideram as necessidades dos utilizadores.

O processo de recolhimento, análise e verificação das necessidades do cliente ou usuário de um sistema é chamado de engenharia de requisitos. O objetivo da Engenharia de Requisitos (ER) é entregar uma exigência de especificação do software correta e completa sem ambiguidades de forma é minimizar os problemas de gerenciamento de requisitos

Tais requisitos, segundo Pleeger (2007), são os resultados de todo um trabalho junto ao cliente que definem um acordo sobre quais são as tarefas que devem ser executadas pelo sistema, estes requisitos, apesar do grande esforço a serem completamente entendidos já no início de qualquer projeto, certamente irão mudar ao longo da progressão.

Segundo Hirama (2011), existe uma necessidade mais importante do que discutir as causas relevantes a erros de entendimentos iniciais dos requisitos, mas sim determinar a forma de minimizar os impactos gerados pelas mudanças durante o desenvolvimento.

Segundo os conceitos de Sommerville (2011), quaisquer que sejam os requisitos, estes devem ser escritos com o maior nível de detalhamento possível para que todas as pessoas envolvidas no processo possam entender. Não obstante, é importante ressaltar que nas condições normais de projeto, existem diferentes grupos de indivíduos envolvidos na engenharia de requisitos para o processo, são usuários, especialistas em aplicações, clientes, gerentes de projeto e engenheiros de hardware, estes últimos atuam na especialidade cuja designação ficou conhecida como a de desenvolvedores.

2.1 Requisitos funcionais

Um requisito funcional deve delinear o que o sistema deve fazer, portanto, de qual forma devem reagir às condições de entrada e a maneira para atender as exigências e consequentemente transformar entradas em saídas (Gava, 2011).

Se os requisitos funcionais são vistos como uma espécie de resposta de um sistema

aos estímulos de entrada, certamente irão executar, segundo Pfleeger (2007), as tarefas as que lhe forem confiadas independentes de quais a linguagens de programação forem utilizadas ou em quais bases de hardware o sistema estiver embarcado.

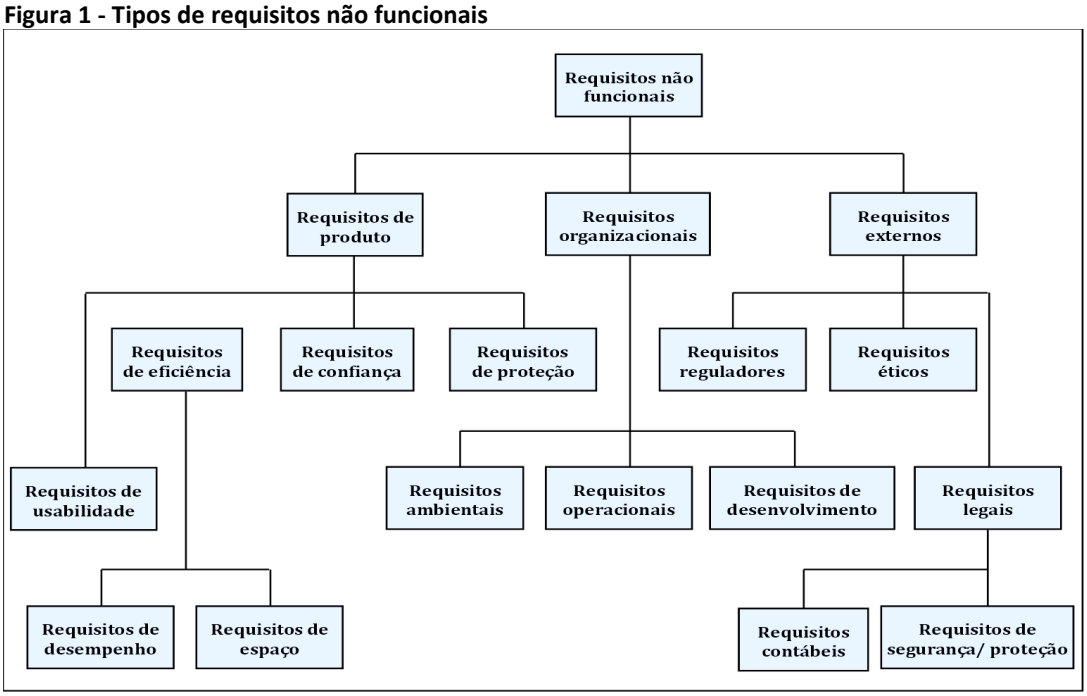
As diversas questões relativas aos requisitos funcionais que serão oferecidos pelo sistema ao usuário devem ser amplamente consistentes, logo não admitindo quaisquer condições contraditórias. Entretanto, Sommerville (2011), numa ótica prática, afirma que quando os sistemas passam a ter maior volume, tornam-se também proporcionalmente complexos e tendem por consequência gerar a impossibilidade em alcançar a plenitude em termos de consistência de requisitos.

2.2 Requisitos não funcionais

Para Pfleeger (2007), os requisitos não funcionais estão diretamente relacionados com as restrições do sistema, portanto impondo limitações às propostas de solução para um problema. Estas condições definem a capacidade e o comportamento do sistema, tais requisitos, por definição do próprio nome não estão relacionados de forma direta com os serviços que se propõem a executar. Neste contexto Sommerville (2011), vai além ao definir que os requisitos não funcionais podem estar relacionados com as características que são emergentes do sistema, como confiabilidade, tempo de resposta ou emprego de área e, portanto são provenientes de peculiaridades requeridas para o software como produto, a organização que o produz e os fatores externos.

Como forma de se classificar os requisitos não funcionais, verifica-se segundo Sommerville (2011), demonstrado na figura 1 a seguir, como se desenvolvem esses encadeamentos.

Figura 1 - Tipos de requisitos não funcionais



Fonte: Sommerville, 2011

3 METODOLOGIA

A condução da presente pesquisa firma-se na abordagem de literaturas do tema em questão, constitui de análises e suas argumentações no que se juntam propósitos à avaliação de técnicas consagradas ao desenvolvimento do software clássico, semelhanças e ou contrastes dos conceitos aplicáveis aos sistemas destinados a soluções embarcadas.

O caráter da exploração da matéria documenta o problema de pesquisa e suas relevâncias. No entanto, tais pressupostos estão aqui delimitados pela apreciação no conjunto de ferramentas necessárias ao desenvolvedor de softwares embarcados e os desafios enfrentados nos níveis de produção e a garantias de funcionalidade.

As fontes de informação cujas características somaram os objetivos para o presente trabalho, são apresentadas na tabela 1.

Tabela 1 – Relação de obras, conteúdos de informações.

Conteúdo relevante	Pertinência	Pesquisados	Utilizados
Engenharia de software	Desenvolvimento de software tradicional	12	5
Métricas de software	Métricas de software em geral	26	4
Métricas de software	Métricas de software embarcado	15	3
Análise de requisitos	Fundamentação para uso de métricas	21	9
Qualidade de software	Fundamentação para análises de distinção	11	6
Produtividade de software	Fundamentação para análises de distinção	7	3

Fonte: Elaborada pelos autores

4 RESULTADOS E DISCUSSÕES

As métricas de software compõem formas de medir características de software que possam ser quantificáveis. Segundo Sommerville (2011), medição é a avaliação necessária para auxiliar a responder em quais aspectos podem-se evidenciar progressos no âmbito geral dos objetivos, de modo a alcançar consistência com as metas desejadas.

Pressman e Maxim (2016, p. 653) afirmam que “Você pode usar medidas para compreender melhor os atributos dos modelos criados e para avaliar a qualidade dos produtos ou sistemas construídos”. No entanto os autores advertem que avaliações de software, podem apresentar dificuldades nas medidas diretas como outras grandezas da física tradicional.

Pode-se concluir a partir destas definições que as medidas, embora que sejam aplicadas de forma indireta constitui uma importante propriedade para o modelamento e análise qualitativa, dentro dos aspectos que envolvem a engenharia de desenvolvimento dos sistemas ou do produto.

4.1 A qualidade de software

Os atributos de qualidade estão diretamente ligados aos processos que demandam o desenvolvimento que qualquer bem de consumo. A importância da qualidade como

característica do produto é compreendida pelas indústrias de manufatura desde seus primórdios, no entanto em termos de desenvolvimento de software decorrem muitos defeitos apesar dos conceitos, regras e moldes específicos (Hirama, 2011).

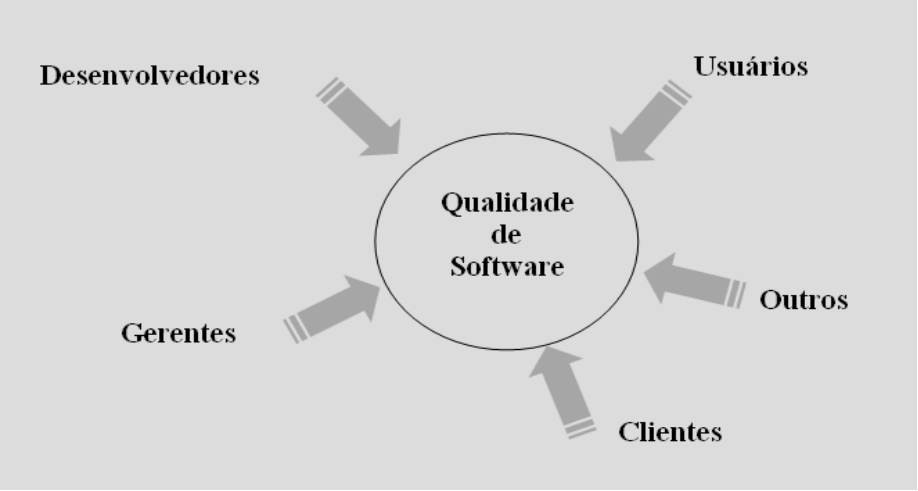
Com bases nestas argumentações, pode-se perguntar “O que significa qualidade de software?”.

Em resposta Hirama (2011) compreende que qualidade deve estar diretamente baseada nas perspectivas dos *stakeholders*, e propõe “Do ponto de vista dos desenvolvedores a qualidade pode ser vista como entendimento a métodos e padrões de software”.

Segundo a afirmação do autor, existe uma evidente necessidade que o desenvolvedor exerça domínio de conhecimento das metodologias e padronizações para o entendimento da qualidade de software.

A seguir a figura 2 ilustra a meta de qualidade no foco das pessoas envolvidas.

Figura 2 - As perspectivas dos *Stakeholders* sobre qualidade de software.



Fonte: Hirama (2011).

4.2 As métricas para a qualidade de software

As métricas de qualidade de software são um subconjunto de métricas de software que se concentra nos aspectos da qualidade do produto, processo ou do projeto. Métricas do produto descrevem as características do software propriamente dito, tais como tamanho, complexidade, características de operabilidade, desempenho e nível de qualidade.

Em relação às métricas de produto, Sommerville (2011), sugere por afirmação de que estas se dividem em dinâmicas e estáticas, neste aspecto este autor as classifica da seguinte forma:

a) As métricas dinâmicas têm relação direta com o funcionamento do programa em execução e os dados são coletados se referem, por exemplo, com o tempo ou o relatório de falhas de computação;
b) As métricas estáticas que por sua vez classificam aspectos do sistema tais como o programa ou a documentação gerada por este.

Em concordância Sommerville (2011), entende que:

> Métricas dinâmicas ajudam a avaliar a eficiência e a confiabilidade de um programa. Métricas estáticas ajudam a avaliar a complexidade, a compressibilidade e a manutenibilidade de um sistema ou componentes de um sistema de software. (Sommerville, 2011).

A partir destas definições, conclui-se que as duas medidas estão correlacionadas, mas de certa maneira elas são diferentes o suficiente para merecer uma análise de maior critério em função dos contextos. As métricas dinâmicas demonstram ter uma grande afinidade com os atributos relativos ao desempenho do software, enquanto que as métricas estáticas produzem maior efeito nas bases de operação dos desenvolvedores do software.

Segundo Pressman e Maxim (2016), uma análise positiva pode ser valiosa, as métricas de software só serão proveitosas se forem caracterizadas como eficazes e seu valor for devidamente comprovado.

De toda forma, relatórios de resultados orientados por requisito poderão mostrar qual porcentagem de teste é atribuída individualmente a cada requisito, isto auxilia na identificação de exageros ou insuficiências destas condições e assim, escalar execuções de teste adequadamente.

Avaliar a qualidade do software não se trata de apenas uma estratégia técnica, salienta Gomes (2010), que a importância de avaliar a qualidade do software embarcado é também uma questão competitiva para quem comercializa este tipo de produto, o autor defende também que recursos e ferramentas amoldadas aos propósitos de excelência podem garantir o sucesso às empresas deste setor.

O software embarcado geralmente precisa ser executado em recursos de hardware limitados (capacidade de processamento, quantidade de memória, número de entradas e saídas físicas, etc.) sendo comum que haja grande esforço para extrair ampla eficiência do mesmo. No entanto, na opinião de Corrêa (2011), é possível que ocorram relaxamentos nos valores qualidade, o que difere das tradicionais práticas normalmente adotadas pela engenharia de software.

É de extrema importância ter um bom conhecimento dos requisitos ditos não funcionais no desenvolvimento de projetos de sistemas embarcados, ainda mais do que na maioria dos projetos de desenvolvimento de software. A rotatividade desses requisitos, ao longo do processo, carece de um rígido controle de qualidade até o final do projeto para evitar retrabalhos que consomem tempo e aumentam os custos (Vieira, 2015).

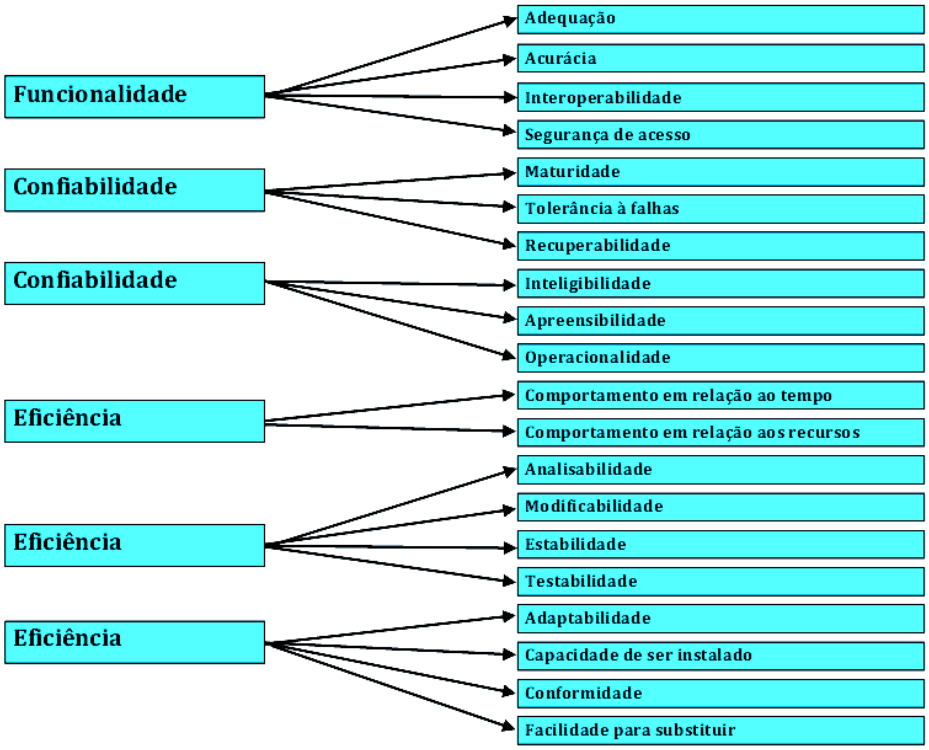
4.3 Norma ISO 9126 (ISO/IEC 25010)

Com referência a norma técnica internacional, Pressman e Maxim (2016), descrevem a *International Organization for Standardization* (ISO) mais exatamente à ISO 9126 como um conjunto de características que têm impacto na competência do software de sustentar o nível de desempenho dentro das normas de qualidade estabelecidas e identificadas segundo os atributos a seguir:

a) Funcionalidade satisfaz os requisitos funcionais do software;
b) Confiabilidade, manutenção do nível de desempenho especificado;
c) Usabilidade, facilidade de uso;
d) Eficiência, relação desempenho / recursos usados;
e) Manutenibilidade, facilidade para fazer alterações;
f) Portabilidade, facilidade de ser portado para outros ambientes.

Para exemplificar os itens relativos à norma ISO 9126 com um enfoque ainda mais objetivo, Pfleeger (2007), propõe uma ilustração hierárquica como mostra a figura 3.

Figura 3 - Modelo de qualidade da norma ISO 9126.



Fonte: Pfleeger (2007).

No entendimento de Pfleeger (2007), o modelo aqui ilustrado revela as características relacionadas do lado direito como sendo referentes à visão do usuário do software, enquanto os quadros da esquerda têm um foco direcionado ao desenvolvedor.

Apesar de uma série de técnicas com o objetivo de medir a qualidade do software, a conclusão de Pressman e Maxim (2016), é que quaisquer dessas formas somente poderão evidenciar uma medida indireta, ou seja, o que realmente pode ser medido é o reflexo ou influencia na ação da qualidade.

4.4 As métricas para a produtividade versus qualidade de software

Em termos de processos da engenharia e desenvolvimento de software, Wazlawick (2019), define como métricas de produtividade, um conjunto de esforços que podem ser empregados para examinar o progresso dos propósitos e possíveis desvios.

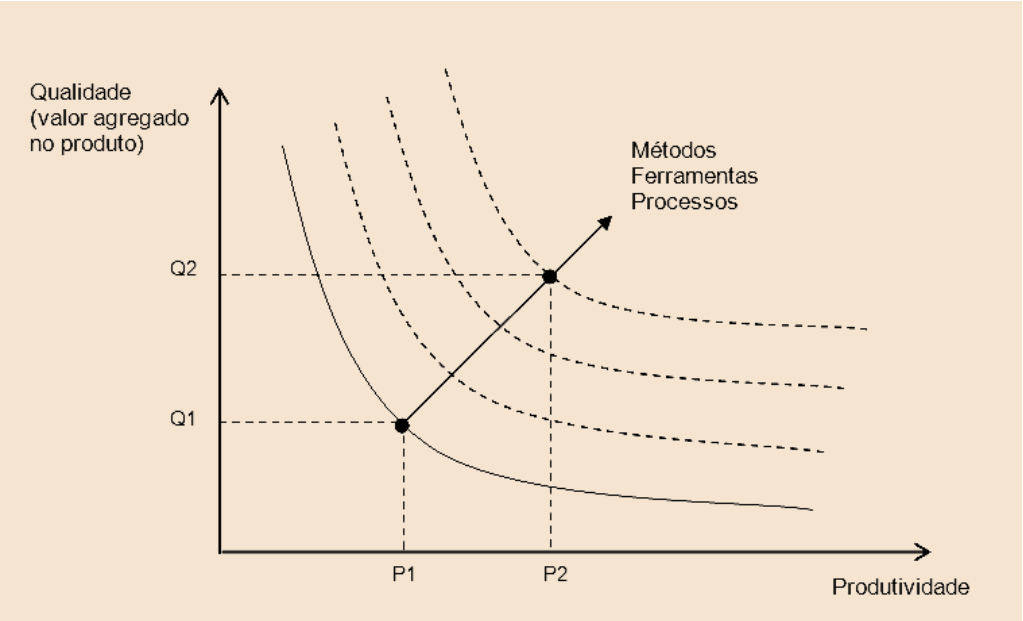
A produtividade consiste de importante estratégia competitiva em qualquer ramo produtivo. Para a indústria de software isto não é diferente, todavia segundo Hirama (2011), não há possibilidades de se alcançar índices almejados de produtividade tão somente com pessoas e processos adequados.

Se o produto prima pela qualidade, este certamente combina alto valor agregado, entretanto não se pondera os índices legados a produtividade. Em outros termos, avaliam-se bons índices de produtividade, porém estes podem comprometer seriamente os níveis de qualidade (Hirama, 2011).

Abaixo, a figura 4, demonstra a relação existente entre os índices de qualidade, produtividade, e o valor que se agrega ao produto. Nota-se que o equilíbrio ideal está

localizado em um ponto médio da curva (P1, Q1), mas neste ponto a qualidade e a produtividade representam índices baixos. Deste modo melhores métodos, ferramentas e processos deslocam a curva para um afastamento dos eixos elevando-se os níveis de produtividade e qualidade (Q2, P2).

Figura 4 - Relação entre qualidade e produtividade.



Fonte: Hirama (2011).

4.5 Planos de desenvolvimento do processo produtivo

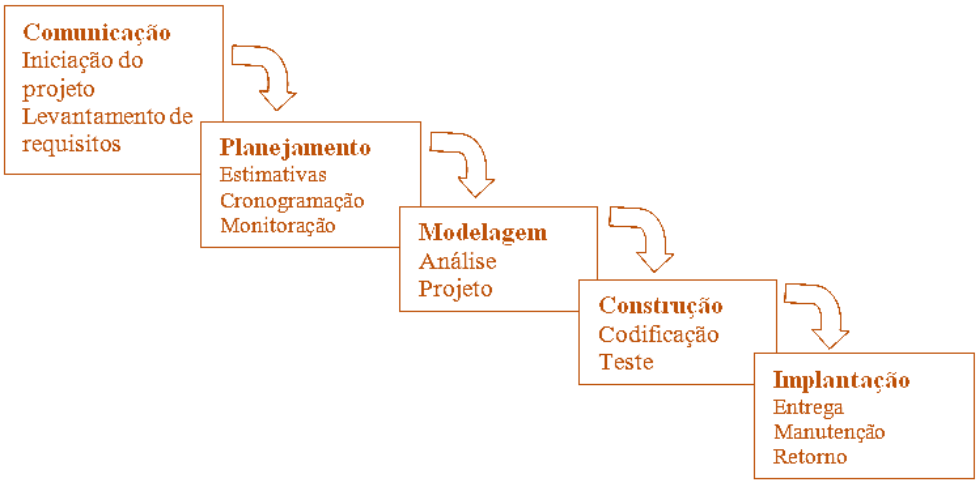
Um gerente de projeto, função por vezes ocupada pelo desenvolvedor, deve estabelecer criteriosamente uma estimativa a três coisas conforme apresentado por Pressman e Maxim (2016), o tempo para o desenvolvimento do projeto, quais serão os níveis de esforço exigidos e quantas pessoas deverão estar envolvidas.

Neste aspecto Sommerville (2011), ainda mais enfático, ao afirmar que para evitar falhas no processo, os gerentes de projeto devem usar um plano de suporte as decisões como forma de medir o progresso, bem como a identificação de riscos para o projeto, sem esquecer a importância nas definições de recursos disponíveis, a divisão do trabalho e um cronograma de tarefas.

4.5.1 Desenvolvimento com base no modelo em cascata

O modelo em cascata (figura 5) descreve um procedimento de incremento que é linear e sequencial, logo quando uma fase de desenvolvimento é completada, prossegue-se para a próxima etapa não havendo, portanto recuo. Segundo Pressman e Maxim (2016), o modelo em cascata tende a inflexibilidade quando há frequência nas mudanças de requisitos, mas constitui de modelo útil quando os requisitos são fixos e permanecem lineares durante todo o ciclo de vida do projeto.

Figura 5 - Modelo de processo em cascata.



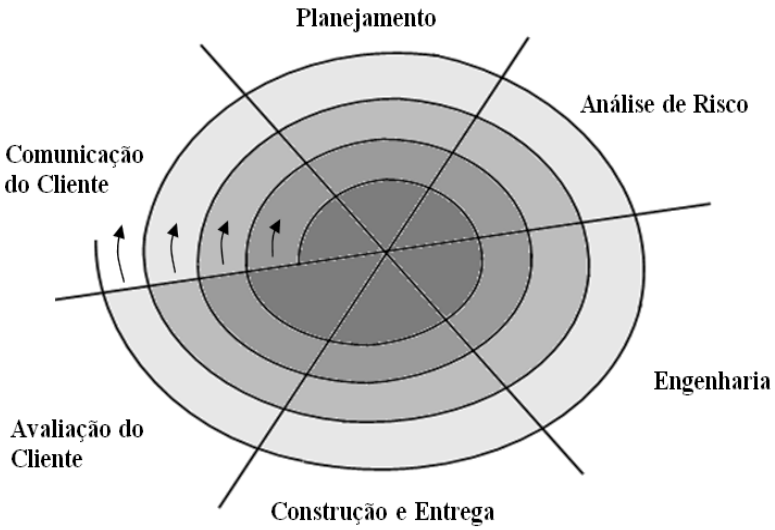
Fonte: Adaptado de Pressman e Maxim (2006).

4.5.2 Desenvolvimento com base no modelo em espiral

Segundo Hirama (2011), a natureza dos processos evolutivos converge diretamente ao progresso incremental de desenvolvimento do produto, propiciando ao desenvolvedor uma visão direta de quaisquer riscos técnicos e gerenciais em todas as fases do projeto.

Hirama (2011) também descreve uma representação de processo evolutivo através de uma variação do exemplo proposto por Barry Boehm (1988), como mostra a figura 6. Este modelo desenvolve o entendimento linear e sequencial, assim como o exemplo cascata, mas conta com a importante propriedade iterativa do processo.

Figura 6 - Processo em espiral.



Fonte: Hirama (2011).

O modelo em espiral emprega a prototipagem como mecanismo de redução ao risco, mais do que isso constitui importante aliado ao desenvolvedor de software, pois permite aplicar um estudo interpretativo da prototipagem em qualquer etapa do projeto, além de manter o método sistêmico do passo a passo típico dos modelos produtivos clássicos, (Pressman e Maxim, 2006).

4.5.3 Desenvolvimento com base no modelo de métodos ágeis

Métodos Ágeis baseiam-se em técnicas, valores e princípios concentrados para orientar e melhorar a forma como as equipes de desenvolvimento de software trabalham; “O que diferencia os Métodos Ágeis dos outros métodos (também conhecidos como tradicionais ou prescritivas) é o enfoque maior nas pessoas e não em processo, e o seu conjunto de valores, princípios e práticas.” (Prikladnicki; Willi; Milani, 2014, p. 10).

Wazlawick (2019) descreve que os propósitos de utilização do modelo, ganham força em 2001 com o lançamento do Manifesto para o Desenvolvimento Ágil de Software, onde dezessete profissionais ligados à área, reuniram-se com o propósito em discutir um equacionamento às metodologias ágeis já anteriormente utilizadas, bem como alternativas às tradicionais técnicas de desenvolvimento de software e aponta no manifesto, os quatro objetivos e os doze princípios respectivamente apresentados na figura 7 e figura 8.

Figura 7 - Os objetivos dos métodos ágeis.

Valores	Valoriza-se
Indivíduos e interações	Sobre processos e ferramentas
Software funcional	Sobre uma documentação abrangente
Colaboração do cliente	Em vez da negociação de contratos
Responder à mudança	Ao invés de seguir um plano

Fonte: Adaptado de https://agilemanifesto.org/.

Figura 8 - Os princípios dos métodos ágeis.

Princípios definidos no Manifesto Ágil
Nossa maior prioridade é satisfazer o cliente através da entrega rápida e contínua de software com valor.
Mudanças nos requisitos são bem-vindas, mesmo nas etapas finais do projeto. Processos ágeis usam a mudança como um diferencial competitivo para o cliente.
Entregar software frequentemente, com intervalos que variam de duas semanas a dois meses, preferindo o intervalo mais curto.
Administradores e desenvolvedores devem trabalhar juntos diariamente durante o desenvolvimento do projeto.
Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte e confie que eles farão o trabalho.
O meio mais eficiente e efetivo de tratar a comunicação entre/para a equipe de desenvolvimento é a conversa “face a face”.
Software funcionando é medida primordial de progresso.
Processos ágeis promovem desenvolvimento sustentável. Os financiadores, desenvolvedores e usuários devem ser capazes de manter o ritmo indefinidamente.
Atenção contínua a excelência técnica e bom design melhoram a agilidade.
Simplicidade – a arte de maximizar a quantidade de um trabalho não feito – é essencial.
As melhores arquiteturas, requisitos e projetos surgem de equipes auto-organizadas.
Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então ajusta seu comportamento de acordo.

Fonte: Adaptado de https://agilemanifesto.org/.

Alguns modelos utilizam várias técnicas e conceitos dos métodos ágeis, inclusive na gestão e desenvolvimento de software destacam-se:

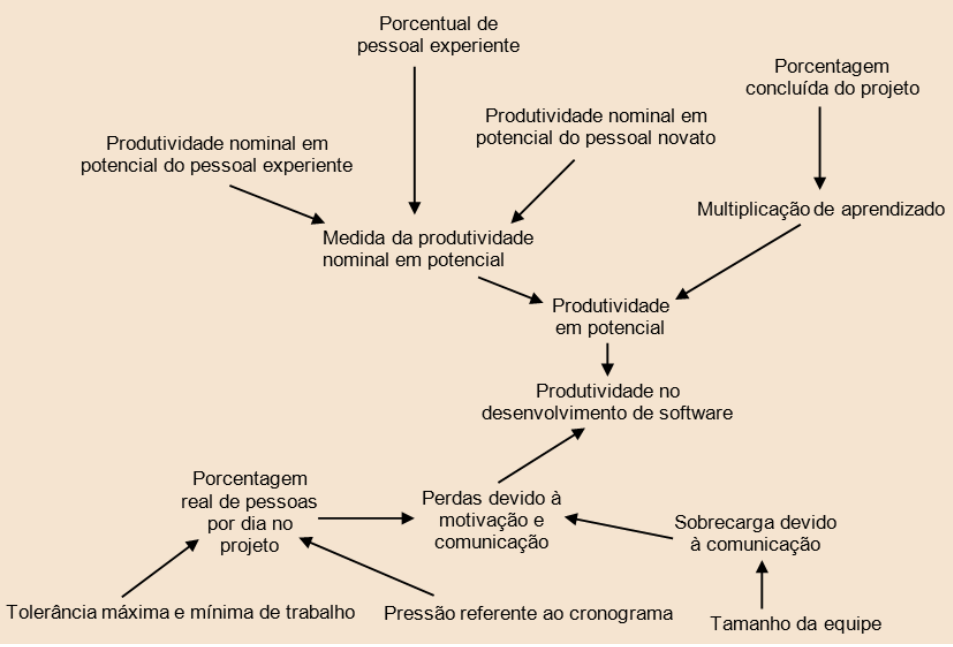
- *Scrum*: Consiste basicamente em um processo onde as decisões são baseadas na observação, experiência e experimentação.
- *Kanban*: Técnica que permite que o software seja desenvolvido em um grande ciclo de desenvolvimento incremental.
- *Extreme Programming* (XP): Defende a comunicação, a simplicidade, a opinião, a coragem e o diálogo entre a equipe de desenvolvimento e o cliente.
- *Lean*: Visa aumentar a eficiência do processo e a qualidade do produto, minimizando custos e desperdícios, uma espécie de 5S.

4.6 Considerações ao modelo dinâmico

Para Pfleeger (2007), uma propriedade altamente desejável em modelos de processo é que o mesmo traduza a forma como se desenvolvem as dinâmicas dos recursos nas atividades até se decomponham em saídas.

O modelo dinâmico atribuído a sistemas de software pode, por exemplo, servir para exemplificar como o desenvolvimento do software tem efeitos na produtividade e como as diversas atividades dos desenvolvedores que envolvem tempo, podem ser estabelecidas em forma de descritivo, (Pfleeger, 2007). A figura 9 demonstra, na visão do autor o exemplo de como a construção do software afeta a dinâmica de produtividade.

Figura 9 - Modelo dos fatores que contribuem para a produtividade.



Fonte Pfleeger (2007).

A fim de esclarecer as relações entre a produtividade de software e pessoal envolvido no projeto, Pfleeger (2007), descreve que:

a) Pode-se observar que a media da produtividade nominal é afetada pela produtividade do pessoal experiente, pela porcentagem do pessoal

experimentado e também dos novatos.

b) Concomitantemente, quanto maior a porcentagem concluída do projeto, maior o aprendizado do pessoal novato, o que tende a aumentar a produtividade em potencial, logo se acrescenta produtividade ao desenvolvimento de software.
c) Por outro lado à pressão sobre o cronograma aliado ao tempo diário que cada desenvolvedor pode se dedicar ao projeto, irá definir o número real de pessoas por dia no projeto. Paralelamente, um grande número de pessoas na equipe pode provocar sobrecarga na comunicação.
d) O efeito destes últimos provavelmente irá incidir em prejuízo devido à motivação e comunicação, logo se promovem perdas à produtividade de software.

O modelo dinâmico oferece a vantagem de esboçar uma visão dos efeitos na produtividade, quando da aplicação do gerenciamento aos recursos humanos, planejamento e controle. No entanto, Pfleeger (2007), defende a importância do uso prudente do modelo, pois os resultados devem ser capazes de evidenciar relações consistentemente quantificadas.

O trabalho do desenvolvedor de software embarcado é ainda mais complicado pela tendência crescente das mudanças de funcionalidade e a complexidade do software para o hardware. Nestes aspectos, a produtividade, costuma ser seriamente afetada pela falta de cultura devidamente constituída no entendimento de requisitos, bem como uma documentação bem elaborada de restrições e alternativas (Ossada e Martins, 2010).

5 CONCLUSÃO

As métricas constituem importantes aliadas do desenvolvedor de software, já que em virtude da necessidade de se produzir com qualidade, devem ser adotados métodos que viabilizam os processos de desenvolvimento.

Durante a pesquisa que resultou neste trabalho, foram identificadas várias técnicas de avaliação de processos descritas na literatura. Com a distinção dessas análises, foi possível avaliar seus benefícios e em outros aspectos também suas limitações.

De modo geral existem algumas dificuldades em se apurar dados de modo sintético e objetivo nas literaturas apontadas com o proposto para a execução deste trabalho em virtude do nível ainda mais profundo na abrangência em que os assuntos relevantes são tratados pelos autores.

Neste trabalho a fundamentação teve início nos requisitos do sistema, essenciais em soluções embarcadas que propiciam o suporte ao entendimento, e irão representar em quais bases o desenvolvedor ou a equipe responsável pelo sistema devem fundamentar os relatórios iniciais do desenvolvimento.

Foi possível avaliar que nos sistemas embarcados, a proximidade ao hardware exigirá esforços para avaliações de qualidade do software produzido em atendimento às condições funcionais. No entanto será um desafio ainda maior a apuração das categorias não funcionais do sistema, uma vez que outra característica destes sistemas reside nas necessidades do ambiente do software interagir com os dispositivos periféricos integrados com diversas condições de exigência oriundas da natureza do próprio hardware.

No aspecto específico das métricas de qualidade e produtividade, este estudo evidenciou as exterioridades particulares ou unânimes em suas aplicações, valendo-se de modelos clássicos, mas diferenciando-se de outras literaturas que indicam frequentemente determinados modelos cujas práticas requerem métodos ainda mais técnicos.

REFERÊNCIAS

BOEHM, Barry W. A spiral model of software development and enhancement. **Computer**, n. 5, p. 61-72, 1988.

CORRÊA, Ulisses B. **Aplicação de métricas de software na predição de características físicas de software embarcado.** Dissertação – (Mestrado em Ciência da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre: UFRG, 2011. Disponível em: https://lume.ufrgs.br/handle/10183/28733 Acesso em: 16 jul. 2023.

GAVA, Luiz W. **Requisitos de software & cooperação.** São Paulo: Blucher 2011. 250p. ISBN: 978-85-8039-048-3.

GOMES, Humberto V. **Metodologia de projeto de software voltada ao teste**. Dissertação – (Mestrado em Ciência da Computação) - Universidade Federal do Rio Grande do Sul, Porto Alegre: UFRG, 2010. Disponível em: https://lume.ufrgs.br/handle/10183/27671 Acesso em: 15 jul. 2023.

HIRAMA, Kechi. **Engenharia de software, qualidade e produtividade com tecnologia.** Rio de Janeiro: Elsevier, 2011. 226p. ISBN: 978-85-352-4882-1.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 25010:2011** - Systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models. ISO/IEC, 2011.

OSSADA, Jaime C.; MARTINS, Luiz E. Um estudo de campo sobre o estado da prática da elicitação de requisitos em sistemas embarcados. 2010. **Workshop em Engenharia de Requisitos.** 2010. Disponível em: https://www.semanticscholar.org/paper/Um-Estudo-de-Campo-sobre-o-Estado-da-Pr%C3%A1tica-da-de-Ossada-Martins/588ad1285c6010b9608c853bb94f4d801f4fd125
Acesso em: 16 jul. 2023.

PFLEEGER, Shari Lawrence. **Engenharia de software, teoria e prática**. 2. ed. São Paulo: Pearson Pentice Hall, 2007. 549p.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software, uma abordagem profissional.** 8. ed. Porto Alegre: AMGH, 2016. 968p.

PRIKLADNICKI, Rafael; WILLI, Renato; MILANI, Fabiano. Métodos Ágeis para Desenvolvimento de Software. Porto Alegre: Bookman, 2014. 312p.

SOMMERVILLE, Ian. **Engenharia de software.** 9. ed. São Paulo: Pearson Pentice Hall, 2011. 542p. ISBN: 978-85-7936-108-1.

VIEIRA, Andrws Aires. **Uma abordagem para estimação prévia dos requisitos não funcionais em sistemas embarcados utilizando métricas de software**. Dissertação – (Mestrado em Ciência da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre: UFRG, 2015**.** Disponível em: https://lume.ufrgs.br/handle/10183/117766 Acesso em: 17 jul. 2023.

WAZLAWICK, Raul. **Engenharia de software: conceitos e práticas**. 2. ed. Rio de Janeiro: Elsevier Editora Ltda., 2019. 320p. ISBN: 978-85-3529-272-5.

SOBRE OS AUTORES

_____

[i] **LUIZ CARLOS CANNO**

Graduado em Tecnologia de Automação Industrial (2009) com Especialização em Gestão Empresarial pela Universidade Nove de Julho (2012), e Especialização em Docência na Educação Profissional e Tecnológica pelo SENAI CETIQT (2015). Professor na Faculdade de Tecnologia SENAI do curso de Tecnologia em Eletrônica Industrial e Pós-graduação em Sistemas Embarcados. https://orcid.org/0000-0001-9331-9309

[ii] **LEANDRO POLONI DANTAS**

Engenheiro (2004) e Doutor (2018) em Engenharia Elétrica pelo Centro Universitário FEI. Atuou por 15 anos na indústria eletrônica no desenvolvimento de novos produtos. Desde 2009, vem lecionando em cursos de pós-graduação, graduação e de nível técnico em diferentes instituições paulistanas. Atualmente é professor na Faculdade de Tecnologia SENAI e no Insper. https://orcid.org/0000-0003-3674-336X

[iii] **FERNANDO SIMPLICIO DE SOUSA**

Professor da Faculdade SENAI no curso de Pós-Graduação em Sistemas Embarcados. Mestre em Engenharia Elétrica pela Universidade Federal do ABC (UFABC) e Pós-Graduado (Lato Sensu) pela Universidade Mackenzie. Graduado em Gestão de Pequenas e Médias Empresas pela UNIP e em Projetos Mecânicos pela Faculdade de Tecnologia de São Paulo (UNESP/FATEC-SP). https://orcid.org/0009-0009-5760-4845

[iv] **MARCONES CLEBER BRITO DA SILVA**

Tecnólogo em Mecatrônica Industrial (2011), Engenheiro Mecatrônico (2013) e Especialista em Engenharia de Manutenção Industrial pela Centro universitário Eniac (2013). Mestre em Tecnologia Nuclear (2020) pela Universidade de São Paulo. Desde 2011, vem lecionando em cursos de nível técnicos e de graduação. Atualmente é professor da Faculdade de Tecnologia SENAI e na FESA. https://orcid.org/0000-0002-3690-1682